SC-CoIT: Self-Corrective Interleaved Chain of Thought via Bidirectional Cross-Modal Recovery

Anonymous authors
Paper under double-blind review

Abstract

Unified multimodal models (UMMs) can generate Interleaved Chain of Thought (Interleaved CoT) traces that alternate between language and visual thoughts. Yet correct-only supervised fine-tuning provides no recovery target when an intermediate node fails. We identify two cascade failures: a **Draw Error**, where the text expert proposes a valid action but the image expert misexecutes it, and a **Policy Error**, where the text expert proposes an action inconsistent with the task or environment and the image expert follows it without verification. From ground-truth Interleaved CoT traces, we construct a bidirectional error-recovery training dataset and propose **Self-Corrective Interleaved Chain of Thought (SC-CoIT)**, which trains the modality experts to check each other's outputs and recover from cross-modal errors. For Draw Errors, the text expert grounds the discrepancy and directs a same-step redraw; for Policy Errors, the image expert emits a visual reflection signal, prompting the text expert to regenerate the step. To adapt both recovery branches to failure modes that evolve during training, iterative error replay samples new errors from the current checkpoint on validated prefixes and adds them to subsequent training. Together, these mechanisms improve the model's ability to inspect generated visual states, reject invalid actions before execution, and recover from both error types within the same Interleaved CoT trajectory.

1 Introduction

Unified multimodal models (UMMs) place visual understanding and image generation in one autoregressive sequence (Chameleon Team, 2024; Wang et al., 2024; Xie et al., 2025; Chern et al., 2024; Chen et al., 2025; Deng et al., 2025; Liu et al., 2026; Zhang et al., 2026a; Cui et al., 2025). This enables Interleaved Chain of Thought (Interleaved CoT), in which language and visual thoughts alternate and generated images become reasoning states. Models can render diagrams through external tools (Hu et al., 2024; Fang et al., 2025) or generate visual thoughts directly (Chern et al., 2025; Shi et al., 2025). Zebra-CoT supplies large-scale traces across geometry, physics, spatial reasoning, and strategic games (Li et al., 2025). ThinkMorph, Monet, and Omni-R1 extend the paradigm through generalization, latent reasoning, and reinforcement learning, respectively (Gu et al., 2026; Hu et al., 2026b;a), while WEAVE exposes remaining challenges in multi-turn Interleaved CoT (Wei et al., 2025).

These advances also change how errors propagate. Specifically, standard SFT on correct thought trajectories learns each text or image node from a correct prefix—requiring all prior nodes remain valid. At inference, every generated node becomes the next node's input. One deviation therefore creates an erroneous prefix absent from the training targets. The model has not learned to recognize that unexpected state, prevent the error from propagating, or return to a valid path.

This mismatch appears at two opposing modality boundaries. A ***Draw Error*** occurs when the text expert proposes a valid action but the image expert misexecutes it. As shown in Figure 1(a), the image expert omits required highlights for a valid 4-connected-component request, after which the text expert incorrectly reports 'three separate groups.' A ***Policy Error*** occurs when the text expert proposes an invalid action inconsistent with the environment or task and the image expert follows it without verification. In Figure 1(b), the drawer is already open and should be closed, yet the text

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

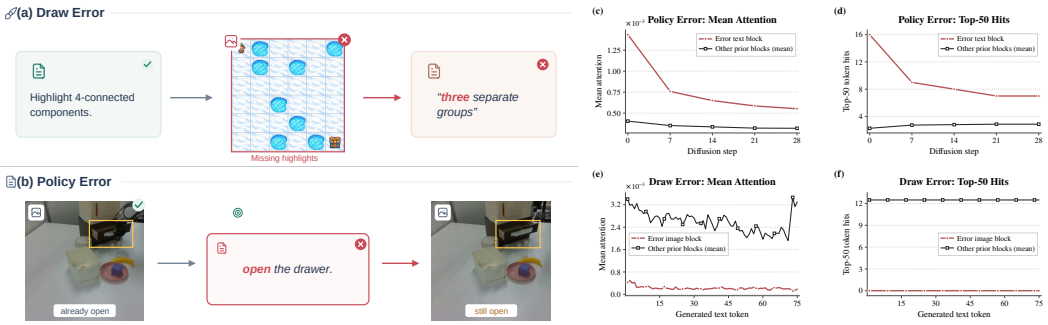

Figure 1: **Cascade failures and their attention signatures.** *(a) Draw Error:* a valid request is followed by an image that omits required component highlights, causing an incorrect count. *(b) Policy Error:* although the open drawer should be closed, the text requests another opening action and the resulting state remains open. *(c,d)* Invalid action text dominates image-denoising attention, whereas *(e,f)* the faulty image is largely ignored during subsequent text generation.

expert requests 'open the drawer' and the image expert leaves it open. Each local error thus becomes a state on which later steps depend.

Figure 1(c–f) reveals distinct attention patterns. For Policy Errors, invalid action text dominates mean attention and Top-50 hits during image denoising (c,d), indicating that the image expert follows the invalid action without verification, a form of over-compliance with text instructions. For Draw Errors, the faulty image receives lower mean attention during subsequent text generation (e), with near-zero Top-50 hits versus about 12 for other prior blocks (f). The text expert therefore bypasses the latest image and relies on earlier language history. We call this behavior *information hacking*; it reflects underuse of the current visual state.

These patterns show that recovery requires cross-modal verification: the image expert checks text actions before execution, and the text expert inspects the latest visual state. Although self-correction without external feedback is limited by error detection and feedback quality (Huang et al., 2024; Tyen et al., 2024; Wu et al., 2025), Interleaved CoT exposes each expert's output to the other. In a Mixture-of-Transformers (MoT), modality-specific paths share this context, enabling same-step detection and recovery when errors become visible.

We therefore introduce **Self-Corrective Interleaved Chain of Thought (SC-CoIT)** and a bidirectional error-recovery dataset built from ground-truth Interleaved CoT traces. After a validated prefix, we sample a candidate node and use an offline VLM to flag violations of the instruction, task state, or environment. An erroneous node remains in context but is not an imitation target; reflection or rejection, a same-step recovery target, and the correct suffix follow. A paired valid branch teaches when to continue, turning failed trajectories into local, supervised cross-modal recovery events.

SC-CoIT has two branches. Reflect-Draw-SFT trains the text expert to inspect a faulty image, preserve reliable content, and direct a same-step redraw. Reflect-Policy-SFT trains the image expert to emit a visual reflection signal for an invalid action, prompting a textual retry before corrected execution. To adapt both recovery branches to evolving model errors, SC-CoIT uses iterative error replay to refresh their supervision with new failures from the latest checkpoint. At inference, reflection signals and retries remain in the ordinary sequence, without an external VLM, detector, or controller.

We make the following contributions:

1. **Recovery-oriented supervision for Interleaved CoT.** We construct a bidirectional error-recovery dataset that pairs intermediate text and image failures with cross-modal feedback, same-step recovery, and valid counterparts. This supervision teaches the model when to correct and when to continue.
2. **Self-Corrective Interleaved Chain of Thought.** We propose SC-CoIT, a unified framework in which text and image experts correct each other's outputs within a reasoning trajectory. One path diagnoses and redraws faulty images. The other rejects invalid text actions

and retries the step. Iterative replay updates supervision as failures evolve, with no external evaluator at inference.

3. **Empirical evidence for cross-modal recovery.** Experiments across diverse visual reasoning tasks show that SC-CoIT improves interleaved reasoning at both modality boundaries. Ablations establish complementary recovery roles, and attention analysis finds stronger grounding in faulty visual states during correction.

2 Related Work

Unified Multimodal Models (UMM). Unified multimodal models combine visual understanding and image generation within one architecture. Autoregressive systems such as Chameleon, Emu3, Show-o, and Janus-Pro establish this model family (Chameleon Team, 2024; Wang et al., 2024; Xie et al., 2025; Chen et al., 2025), while BAGEL scales it through interleaved pretraining (Deng et al., 2025). Recent work reduces interference between comprehension and synthesis through unified tokenizers, decoupled diffusion decoders, and aligned representations (Liu et al., 2026; Zhang et al., 2026a; Wang et al., 2026a;b), while post-training improves visual tool use and learns when or what to draw (Zheng et al., 2025; Wang et al., 2025; Gao et al., 2026; Yang et al., 2026; Wu et al., 2026). These advances strengthen multimodal generation and reasoning but largely treat generated images and proposed actions as valid inputs to subsequent steps. SC-CoIT instead learns to recover from image-generation and policy errors at text–image boundaries.

Interleaved Chain of Thought. In Interleaved CoT, models alternate between text thought blocks and visual thought generation during inference. Zebra-CoT is the foundational dataset for this line of work, providing 182K samples across geometry, physics, algorithms, spatial tasks, and strategic games (Li et al., 2025). These tasks benefit from sketching or visualizing intermediate states as a natural reasoning aid. ThinkMorph conducts a systematic study of the emergent properties of Interleaved CoT: models trained on interleaved multimodal chains of thought outperform text-only or image-only variants on out-of-distribution benchmarks and benefit from test-time scaling via diversified thought generation (Gu et al., 2026). Monet moves visual thoughts into the latent space, enabling continuous embedding generation as intermediate reasoning steps without discrete image decoding (Hu et al., 2026b). Omni-R1 introduces a two-stage SFT-then-RL training pipeline for unified generative reasoning across diverse multimodal tasks (Hu et al., 2026a). WEAVE shows that in-context interleaved comprehension and generation remains harder than single-turn tasks and provides benchmarks for multi-turn visual communication with a UMM (Wei et al., 2025). Existing methods primarily learn from correct Interleaved CoT trajectories and do not explicitly train recovery from errors produced along the model's own inference traces.

Cross-Modal Reflection. Reliable self-correction requires both error detection and useful feedback. Reflexion and Self-Refine revise outputs through feedback (Shinn et al., 2023; Madaan et al., 2023), yet models correct known errors more reliably than they locate them (Tyen et al., 2024), and correction without external feedback can reduce accuracy (Huang et al., 2024). VISCO likewise finds human critiques helpful but model-generated critiques potentially harmful (Wu et al., 2025), whereas structured verification can improve semantic grounding (Liao et al., 2025). Cross-modal reflection instead uses one modality to inspect the other: Do-You-See checks generated images against text (Zhang et al., 2026b). DIVA and LatentUMM couple understanding and generation through representation differences or dual objectives (Lu et al., 2026; Luo et al., 2026), although improvements need not transfer between the two paths (Kang et al., 2026). DreamSync and DeGF use one path to improve outputs from the other (Sun et al., 2025; Zhang et al., 2025). Critic-V separates reasoning and critique (Zhang et al., 2024), while BAGEL provides modality-specific paths within a unified model (Deng et al., 2025). SC-CoIT trains these paths for bidirectional recovery within interleaved trajectories: the text expert diagnoses faulty images and guides their redraw, while the image expert emits a visual reflection signal for invalid text actions, triggering text-policy correction.

3 Method

Figure 2 presents the overall SC-CoIT framework. Its central idea is to incorporate recoverability into the learning objective for Interleaved CoT: an intermediate error remains part of the current context and is paired with a valid subsequent correction as training supervision, making recovery

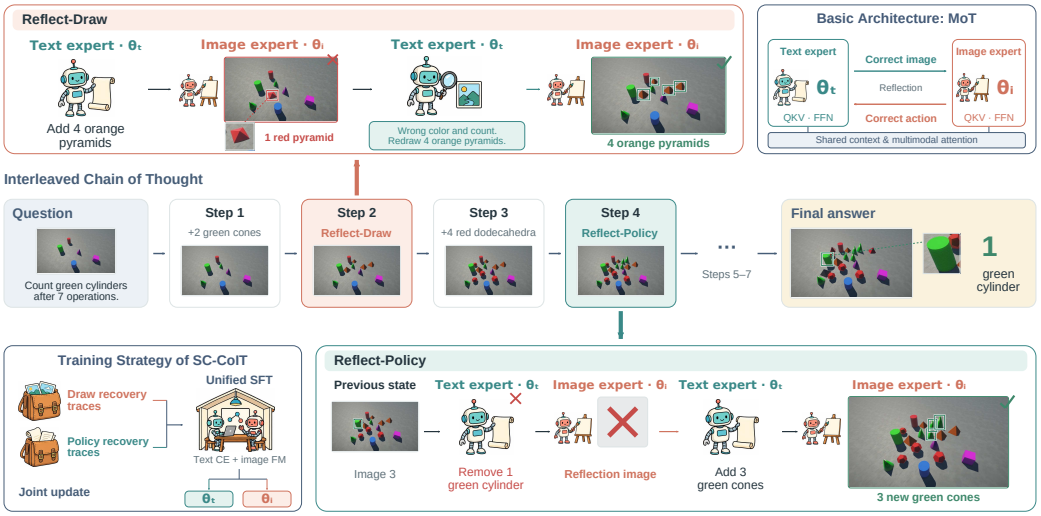

Figure 2: **Self-Corrective Interleaved Chain of Thought.** Recovery is inserted directly into an Interleaved CoT trace. Reflect-Draw-SFT uses textual reflection to diagnose a faulty visual thought and redraw the same step, whereas Reflect-Policy-SFT uses a visual reflection signal to reject an invalid action and retry before execution. In the MoT architecture, modality-specific text and image paths share the interleaved context and cross-check each transition. Draw- and policy-recovery traces jointly supervise both paths with text cross-entropy and image flow-matching losses.

a capability of the interleaved reasoning process itself. We first formalize interleaved trajectories and the positions at which errors arise, then introduce the corresponding recovery methods, unified training objective, and iterative error replay.

3.1 Problem Formulation

A ground-truth Interleaved CoT trace for problem P is

$$\tau^{\mathrm{gt}} = (P, T_1^{\mathrm{gt}}, I_1^{\mathrm{gt}}, \ldots, T_N^{\mathrm{gt}}, I_N^{\mathrm{gt}}, A) = \tau_{<i}^{\mathrm{gt}} \oplus (T_i^{\mathrm{gt}}, I_i^{\mathrm{gt}}) \oplus \tau_{>i}^{\mathrm{gt}}. \tag{1}$$

Here, N is the number of logical steps, A is the final text answer, and $\oplus$ denotes sequence concatenation. T_i is the textual thought at logical step i and contains the observation, reasoning, and action that drives the next visual state. I_i is the visual thought generated at the same step. The prefix $\tau_{<i}^{\mathrm{gt}}$ ends immediately before T_i^{gt} and contains the latest validated task state at the start of the step, while the suffix $\tau_{>i}^{\mathrm{gt}}$ begins immediately after I_i^{gt}. The index i always denotes a logical task step, and inserted recovery nodes do not advance it. The superscript gt marks ground-truth nodes.

Each logical step performs a cross-modal state transition. The textual thought T_i proposes the next action from the current validated state, and the image generation path realizes that action as the visual thought I_i. The text policy determines how the state transitions, while image generation determines how that transition is realized. Errors can therefore enter the trajectory at two interfaces: a *Policy Error* occurs at the textual node before rendering, whereas a *Draw Error* occurs at the image node after the action is executed. A failure at either node changes the conditioning history of all subsequent nodes, making error recovery a necessary capability of interleaved reasoning itself.

If the action proposed in T_i is not executable under the current validated state and the task rules, the textual node constitutes a *Policy Error*. If the action is executable but the generated image I_i does not faithfully realize the corresponding state transition, the image node constitutes a *Draw Error*. We denote the erroneous text action and image by T_i^- and I_i^-, respectively, where the superscript $-$ marks an erroneous generated node retained after error detection. Figure 3 shows the decision boundary and representative categories of the two error types.

Draw Errors include *Instruction Mismatch*, *State Inconsistency*, and *Rendering Error*, whereas Policy Errors include *Unavailable Target*, *Infeasible Action*, and *Unmet Precondition*.

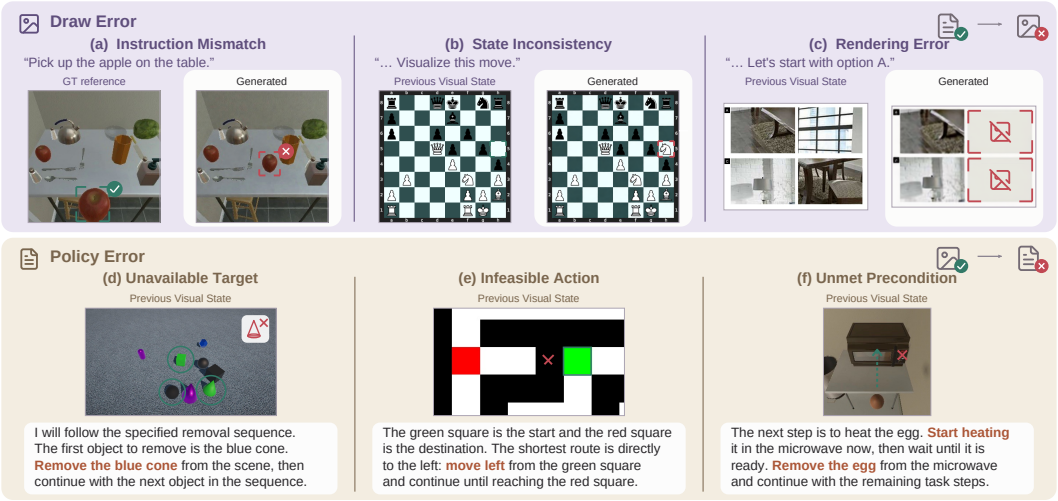

Figure 3: **Draw Error and Policy Error taxonomy.** A Policy Error is a text action that is not executable under the current visual state and task rules, whereas a Draw Error occurs when an executable action is not faithfully realized by image generation. Panels (a)–(c) and (d)–(f) show the three subtypes of each error.

The two error types define two distinct recovery points. A Draw Error retains I_i^- in context and initiates Reflect-Draw, through which the text path diagnoses the visual discrepancy and redraws the current step. A Policy Error initiates Reflect-Policy from T_i^-, generates a visual reflection signal before the task state is updated, and regenerates the text policy. Both paths complete recovery within the same logical step i.

3.2 Reflect-Draw-SFT

Reflect-Draw-SFT learns a cross-modal recovery loop that turns an erroneous image into textual reflection and then a same-step redraw. Rather than bypassing a faulty visual thought and continuing from earlier language context, the text expert inspects the newest image, states how it differs from the valid task state, identifies reliable visual content to preserve, and produces a targeted redraw instruction. The image expert then regenerates the image for the same logical step, turning an observed Draw Error into a supervised recovery event within the original Interleaved CoT sequence.

To construct a recovery trajectory, we sample image node i from a GT Interleaved CoT trace and use $\tau_{<i}^{\mathrm{gt}} \oplus T_i^{\mathrm{gt}}$ as the validated prefix. A model trained on GT Interleaved CoT traces generates one image candidate from this prefix. We use an external VLM judge to assess the candidate against the problem, validated prefix, and current text instruction, checking whether it executes the instruction, preserves task-critical state, and meets generation-quality requirements. Correct candidates are excluded from the error-recovery set, while an incorrect candidate is retained as I_i^-.

For each I_i^-, we use the VLM to produce a first-person reflection that states the visual discrepancy, identifies the reliable content to preserve, and specifies the redraw. We insert this reflection as an ordinary text node, then append the GT recovery image and the original GT suffix:

$$\tau_i^{\mathrm{draw}} = \tau_{<i}^{\mathrm{gt}} \oplus \left(T_i^{\mathrm{gt}}, I_i^-, \text{reflection text}, I_i^{\mathrm{gt}}\right) \oplus \tau_{>i}^{\mathrm{gt}}. \tag{2}$$

Here, I_i^{gt} restores the correct task state instead of invoking an unconstrained resample.

The faulty image I_i^- remains in context but receives no flow-matching loss; the reflection text, recovered image, and GT suffix receive standard supervision under Eq. (4). Each error trajectory is also paired with a correct-image trajectory that shares $\tau_{<i}^{\mathrm{gt}}$ and T_i^{gt}, making the generated image the only changed input to the following text. This contrast makes the continuation depend on the current visual state, directly targets information hacking, and avoids encouraging reflection after correct generations.

3.3 REFLECT-POLICY-SFT

Reflect-Policy-SFT learns a cross-modal recovery loop that turns an erroneous text action into a visual reflection signal and then a same-step retry. Instead of rendering an invalid action, the image expert emits an image-valued signal I^{ref}. In our implementation, I^{ref} is a fixed red-cross image (RedX) on a neutral gray background. The text expert then regenerates the current logical step, and the image expert executes the corrected action. Because the visual reflection signal occupies the expected text-to-image position but contains no task-state geometry, it can reject an action before the task state changes without introducing a separate output channel or representing an alternative world state.

To construct a recovery trajectory, we sample text node i from a GT Interleaved CoT trace and use $\tau^{\text{gt}}_{<i}$, which ends at the latest valid image state I^{gt}_{i-1}, as the validated prefix. A model trained on GT Interleaved CoT traces generates one text candidate from this prefix. Here, we similarly use the VLM to judge the candidate from the visible problem, validated prefix, and current text, checking whether it correctly understands the task, remains consistent with the environment and preceding reasoning, satisfies spatial and physical constraints, and can reach its target state. Correct candidates are excluded from the error-recovery set, while an invalid candidate is retained as T^{-}_{i}.

For each T^{-}_{i}, we insert the fixed visual reflection signal and then append the GT text, GT image, and original GT suffix for the same step:

$$\tau^{\text{policy}}_{i} = \tau^{\text{gt}}_{<i} \oplus \left(T^{-}_{i}, I^{\text{ref}}, T^{\text{gt}}_{i}, I^{\text{gt}}_{i}\right) \oplus \tau^{\text{gt}}_{>i}. \tag{3}$$

The VLM is used only to identify and annotate the error during data construction; the signal is a predefined visual recovery target rather than a VLM-generated task state. Because I^{ref} encodes no new task state, the retry still starts from I^{gt}_{i-1} and remains at logical step i. The recovered image I^{gt}_{i} is the first new task state after I^{gt}_{i-1}.

The invalid text T^{-}_{i} remains in context with zero cross-entropy loss, whereas the visual reflection signal, T^{gt}_{i}, I^{gt}_{i}, and the GT suffix receive standard supervision under Eq. (4). Each error trajectory is also paired with valid-action branches, including semantically equivalent commands and difficult but feasible targets. The model therefore learns to reject invalid actions selectively while rendering valid actions normally rather than using the signal as a default.

3.4 TRAINING STRATEGY

We jointly train on ground-truth traces, correct counterfactual branches, normal valid-action executions, and the two error-recovery branches. Faulty images and actions condition recovery but are not generation targets. The reflection text, visual reflection signal, corrected nodes, and ordinary correct nodes receive supervision. This pattern teaches the model to read an observed fault without reproducing it while preserving normal behavior on valid trajectories. Appendix A gives the node masks in Table 7.

Both paths share one objective. Let $\mathcal{T}$ and $\mathcal{I}$ denote the text-token and image latent-token positions in a packed batch. Binary masks m_T and m_I select the positions that receive cross-entropy and flow-matching supervision:

$$\mathcal{L}(\theta) = \frac{\sum_{j\in\mathcal{T}} m_T(j)\,\mathcal{L}^{(j)}_{\text{CE}}}{\sum_{j\in\mathcal{T}} m_T(j)} + \frac{\sum_{j\in\mathcal{I}} m_I(j)\,\mathcal{L}^{(j)}_{\text{FM}}}{\sum_{j\in\mathcal{I}} m_I(j)}. \tag{4}$$

The index j identifies a local token position. Each denominator counts the supervised positions of its modality. The image term is the feature-averaged mean-squared error on the flow-velocity target. Text CE and image FM use unit weights. The two branches jointly minimize Eq. (4) with shared parameters and differ only in which recovery positions receive supervision.

We first warm up on curated static trajectories, then repeatedly collect errors from the current model. Each verified error is paired with its recovery and a correct counterpart, mixed with historical errors, and replayed together with ground-truth and positive samples. The resulting curriculum follows the model's changing failure distribution instead of relying on errors from a fixed checkpoint: resolved errors become less prominent, while newly exposed and harder errors enter later rounds.

Historical replay preserves earlier recovery behaviors, and the retained positive data anchors normal generation. Both branches update the same model in each round without alternating optimization. All signals evaluate individual modality transitions and optimize Eq. (4). The mixture and replay schedule are detailed in Appendix A and Algorithm 1.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Training. We initialize Self-Corrective Interleaved Chain of Thought from Bagel-Zebra-CoT-7B, a BAGEL checkpoint trained for native Interleaved CoT on Zebra-CoT (Li et al., 2025; Deng et al., 2025). We construct the error-recovery training dataset entirely from the Zebra-CoT training split. No VisWorld-Eval examples enter training. Appendix A reports the trajectory mixture, node-level loss masks, and iterative replay schedule.

Evaluation. We evaluate zero-shot on VisWorld-Eval (Wu et al., 2026). Its 3,946 test examples span five world-simulation tasks: Paper Folding, Multi-Hop Manipulation, Ball Tracking, Maze, and Sokoban. Cube 3-View and MMSI Positional Relationship test world reconstruction. We use the released test examples, prompts, and verifiers without modification. Each task reports answer accuracy. Overall accuracy is computed across all 3,946 test examples. Avg-5 macro-averages the five non-grid tasks and excludes Maze and Sokoban. Avg-7 macro-averages all seven tasks. Table 1 copies baseline scores from the published leaderboard.

4.2 MAIN RESULTS

Table 1: Zero-shot accuracy (%) on VisWorld-Eval. Published leaderboard scores are taken from Wu et al. (2026). Avg-5 excludes Maze and Sokoban; Avg-7 covers all tasks. Bold marks the column best.

Model	Paper	Multi	Ball	Cube	MMSI	Maze	Soko.	Avg-5	Avg-7
Proprietary models									
Gemini 3 Flash	25.60	75.40	**55.30**	52.70	41.30	73.90	**99.30**	**50.00**	**60.50**
Gemini 3 Pro	27.00	74.50	44.70	**53.30**	**49.60**	33.50	90.20	49.80	53.20
Seed 1.8	10.60	75.20	24.40	42.50	38.80	**83.90**	68.30	38.30	49.10
GPT 5.1	6.40	73.90	34.80	44.50	44.80	0.60	62.80	40.80	38.20
o3	13.50	68.10	24.70	37.70	44.40	0.00	36.00	37.60	32.00
Qwen3.7-Plus	12.71	**78.96**	41.11	46.46	41.00	5.42	83.13	44.05	44.11
Open-source models									
Qwen3-VL-8B-Thinking	11.00	49.30	17.80	21.20	27.70	0.00	5.80	25.40	18.90
BAGEL-7B-MoT	11.20	31.60	19.40	26.80	27.20	0.00	0.20	23.20	16.60
BAGEL-Zebra-CoT	11.67	22.92	12.11	22.29	27.20	0.00	0.00	19.24	13.74
Our model									
SC-CoIT (Ours)	**28.75**	42.29	29.88	50.00	49.04	0.00	0.83	39.99	28.69

SC-CoIT more than doubles the accuracy of its BAGEL-Zebra-CoT baseline. Across the evaluation examples, overall accuracy rises from 13.66% to 29.07%, a gain of 15.41 percentage points and a 2.13× improvement. The macro averages show the same pattern: Avg-5 rises from 19.24% to 39.99%, and Avg-7 rises from 13.74% to 28.69%. The gains span all five non-grid tasks, reaching 27.71 points on Cube, 21.84 on MMSI, 19.37 on Multi-Hop, 17.77 on Ball Tracking, and 17.08 on Paper Folding.

Section 4.3 and Appendix C attribute these gains to complementary visual- and policy-error recovery. Grid planning remains weak at 0.00% on Maze and 0.83% on Sokoban, with failures analyzed in Appendix B.

Table 2: Ablation of the two reflection-training branches on VisWorld-Eval.

Model	Paper	Multi	Ball	Cube	MMSI	Maze	Soko.	Avg-5	Avg-7
BAGEL-Zebra-CoT	11.67	22.92	12.11	22.29	27.20	0.00	0.00	19.24	13.74
Reflect-Draw-SFT only	17.92	31.25	25.20	39.38	41.76	0.00	**0.83**	31.10	22.33
Reflect-Policy-SFT only	23.33	38.54	17.77	35.63	41.00	0.00	0.00	31.25	22.32
SC-CoIT	**28.75**	**42.29**	**29.88**	**50.00**	**49.04**	0.00	**0.83**	**39.99**	**28.69**

Table 3: Category-level error detection and immediate repair (%), judged by Qwen3.7-Plus. Quality denotes Draw reflections scoring at least 3/5. Repair/Det. and Repair/All measure next-node recovery over detected errors and all errors, respectively. Policy precision and recall are audit-weighted.

Task	Precision	Recall	Quality	Repair/Det.	Repair/All
Draw reflection					
Ball Tracking	**100.00**	49.79	84.62	1.37	0.68
Cube	97.23	**55.54**	**93.16**	1.71	0.95
MMSI	88.29	35.38	91.84	**17.35**	**6.14**
Multi-Hop	98.61	52.16	86.06	8.08	4.21
Paper Folding	95.88	36.35	91.85	2.58	0.94
Overall	97.47	47.71	88.58	4.80	2.29
Policy reflection (visual reflection signal)					
Ball Tracking	**89.63**	28.83	–	6.67	1.94
Cube	62.80	**54.79**	–	11.82	**6.37**
MMSI	61.77	41.29	–	7.08	3.12
Multi-Hop	84.30	28.32	–	**14.89**	4.11
Paper Folding	62.26	33.46	–	4.72	1.72
Overall	71.02	34.91	–	8.78	3.15

4.3 Ablation Studies

Table 2 isolates the two recovery branches by comparing Reflect-Draw-SFT only, Reflect-Policy-SFT only, and their combination. The single branches reach 22.93% and 21.90% overall accuracy and show complementary task profiles: Reflect-Policy-SFT leads by 5.41 points on Paper Folding and 7.29 on Multi-Hop, whereas Reflect-Draw-SFT leads by 7.43 on Ball Tracking, 3.75 on Cube, and 0.76 on MMSI. Combining both improves all five non-grid tasks over the stronger single branch and reaches 29.07% overall accuracy, 6.13 and 7.17 points above Reflect-Draw-SFT only and Reflect-Policy-SFT only, with the largest margins on Cube (10.62) and MMSI (7.28).

4.4 Category-wise Error Detection and Repair

Table 3 reports category-level detection and next-node repair for Draw reflection and Policy reflection driven by the visual reflection signal.

The category split reveals complementary recovery regimes and a shared post-detection bottleneck. Draw reflection maintains strong diagnosis quality across tasks and repairs MMSI most effectively, where a grounded diagnosis translates naturally into scene regeneration. Its consistently strong diagnosis quality across all categories shows that the text expert can identify visual faults beyond a single task family. Policy reflection is strongest on Cube and Multi-Hop, where the visual reflection signal redirects the reasoning action before the next state is generated. Across both branches, fewer than 9% of detected errors are repaired at the next node, making the conversion from reflection to executable correction the main limitation. The two branches therefore solve different stages of the cascade. Draw reflection repairs visual execution, while Policy reflection corrects the action that drives it, explaining their complementary gains in Table 2.

4.5 Comparison of Policy-Error Reflection Representations

Table 4 compares three representations: prior-state replay repeats the last verified image, Reflect Head uses a learned binary gate to trigger a retry, and the fixed visual signal rejects the action within the image channel. Reflect Head leads on Ball Tracking, showing that binary gating suits localized action violations. The visual signal leads on the other four non-grid tasks and both aggregate measures, favoring modality-native feedback for general policy correction.

Table 4: Comparison of policy-error reflection representations (accuracy, %). The visual reflection signal achieves the strongest Avg-5 and Avg-7.

Representation	Paper	Multi	Ball	Cube	MMSI	Maze	Soko.	Avg-5	Avg-7
Prior-state replay	17.92	36.67	21.48	34.79	33.72	0.00	0.00	28.92	20.65
Reflect Head	17.08	32.08	**30.86**	33.96	32.76	0.00	0.00	29.35	20.96
Visual reflection signal	**23.33**	**38.54**	17.77	**35.63**	**41.00**	0.00	0.00	**31.25**	**22.32**

4.6 Attention to Faulty Images during Reflection

We test whether Reflect-Draw-SFT attends more to the faulty visual state than BAGEL-Zebra-CoT using 276 diagnosis-triggered candidates with complete 50-token continuations. Both models receive the same prefix ending in I_1^- and generate their own continuations. At each token, we average attention over all 28 layers and 28 heads and count I_1^- patches among the global Top-50 attention keys. Across the 50-token window, the mean count rises from about 3.5 to 5.4, an increase of roughly 54%, with the largest gap early in reflection and positive increases of about 20%–73% across all five task families (Figure 4). Thus, Reflect-Draw-SFT refers to the faulty image more often during recovery. Because the models generate different continuations and candidates are selected by diagnostic text, this result is behavioral rather than causal evidence.

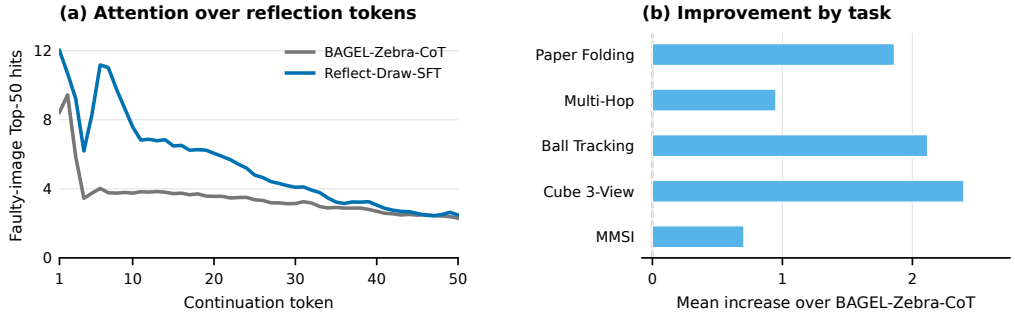

Figure 4: **Faulty-image attention during reflection.** Both panels use 276 Reflect-Draw diagnosis-triggered candidates with complete 50-token continuations. Top-50 hits count faulty-image patches among the 50 highest-attention context keys. (a) Mean hits per continuation token. (b) Mean increases over BAGEL-Zebra-CoT across five task families. Higher values mean faulty-image patches enter the highest-attention keys more often.

5 Conclusion

Reliable Interleaved CoT requires more than correct multimodal trajectories: the model must recover when either modality corrupts the shared state. SC-CoIT turns Draw and Policy Errors into bidirectional, same-step correction targets and refreshes them through iterative replay, enabling recovery without an external evaluator at inference. On VisWorld-Eval, it raises zero-shot accuracy from 13.66% to 29.07% (2.13×); both recovery paths contribute complementary gains, and Reflect-Draw-SFT increases faulty-image Top-50 attention hits by roughly 54%. While Maze and Sokoban expose remaining limits in long-horizon state tracking, these results show that recoverability is a distinct, trainable objective for more reliable multimodal reasoning.

AI USE STATEMENT

In this work, generative AI tools supported code completion, unit-test generation, and language editing. They were not used to formulate research hypotheses or draft the central scientific arguments. The authors independently executed, unit-tested, and verified every AI-generated code snippet. We take full responsibility for the paper's final content and all technical artifacts.

ETHICS STATEMENT

This work uses the existing Zebra-CoT and VisWorld-Eval datasets together with model-generated trajectories. It does not involve human participants or the collection of personally identifiable information. Our evaluation is confined to research benchmarks, and we report the observed failure modes of the resulting model, including its limitations on long-horizon grid planning.

REPRODUCIBILITY STATEMENT

Appendix A documents the data-construction pipeline and prompts, trajectory composition, loss masks, iterative replay schedule, training configuration, and inference settings. The complete source code and configuration files are included in the anonymous supplementary material and will be open-sourced upon publication. Evaluation uses the released VisWorld-Eval examples, prompts, and verifiers without modification.

REFERENCES

Chameleon Team. Chameleon: Mixed-modal early-fusion foundation models. *arXiv preprint arXiv:2405.09818*, 2024. 1, 3

Xiaokang Chen, Zhiyuan Wu, Xingchao Liu, Zizheng Pan, Wen Liu, Zhongang Xie, Xingjun Yu, and Chong Ruan. Janus-pro: Unified multimodal understanding and generation with data and model scaling. *arXiv preprint arXiv:2501.17811*, 2025. 1, 3

Ethan Chern, Jiadi Su, Yanhui Ma, and Pengfei Liu. Anole: An open, autoregressive, native large multimodal model for interleaved image-text generation. *arXiv preprint arXiv:2407.06135*, 2024. 1

Ethan Chern, Zhengqing Hu, Steve Chern, Sihan Kou, Jiadi Su, Yanhui Ma, Ziyu Deng, and Pengfei Liu. Thinking with generated images. *arXiv preprint arXiv:2505.22525*, 2025. 1

Yufeng Cui, Honghao Chen, et al. Emu3.5: Native multimodal models are world learners. *arXiv preprint arXiv:2510.26583*, 2025. 1

Chaorui Deng, Deyao Zhu, Kunchang Li, et al. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*, 2025. 1, 3, 7

Runlin Fang, Chaozhu Duan, Kezhi Wang, Liuyi Huang, Shuo Li, Siyuan Yan, Hongyan Tian, Renyu Zeng, Rui Zhao, Jingren Dai, et al. Got: Unleashing reasoning capability of multimodal large language model for visual generation and editing. *arXiv preprint arXiv:2503.10639*, 2025. 1

Wenxi Gao, Guanxi Lu, Didi Zhu, Hao Mark Chen, Quan Deng, Zhen Wang, Jiankang Deng, and Haoqiang Fan. Model guides you how to draw: Adaptive visual gating for unified multimodal reasoning. *arXiv preprint arXiv:2607.10004*, 2026. 3

Jiawei Gu, Yunzhuo Hao, et al. ThinkMorph: Emergent properties in multimodal interleaved chain-of-thought reasoning. In *International Conference on Learning Representations (ICLR)*, 2026. 1, 3

Yuqian Hu, Wen Shi, Xuhai Fu, Dragomir Roth, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Subhrangsu Krishna. Visual sketchpad: Sketching as a visual chain of thought for multimodal language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 139348–139379, 2024. 1

Zhan Hu, Xu Hu, Yang Liu, Zheng Long, Wei Liu, and Yunzhuo Hao. Omni-R1: Towards the unified generative paradigm for multimodal reasoning. In *Findings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2026a. 1, 3

Zhan Hu et al. Monet: Reasoning in latent visual space beyond image and language. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026b. 1, 3

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations*, 2024. 2, 3

Jiwon Kang, Heeji Yoon, Jaewoo Jung, Jaewon Min, and Seungryong Kim. Transferability between understanding and generation in unified multimodal models. In *European Conference on Computer Vision (ECCV)*, 2026. 3

Ang Li, Charles Wang, Deqing Fu, Kaiyu Yue, Zikui Cai, Wang Bill Zhu, Ollie Liu, Peng Guo, Willie Neiswanger, Furong Huang, Tom Goldstein, and Micah Goldblum. Zebra-CoT: A dataset for interleaved vision language reasoning. *arXiv preprint arXiv:2507.16746*, 2025. 1, 3, 7

Yuan-Hong Liao, Rafid Mahmood, Sanja Fidler, and David Acuna. Can large vision-language models correct semantic grounding errors by themselves? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14667–14678, 2025. 3

Zhewei Liu, Yixiao Ren, et al. TUNA: Taming unified visual representations for native unified multimodal models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. 1, 3

Renjie Lu, Xulong Zhang, Xiaoyang Qu, Shangfei Wang, and Jianzong Wang. DIVA: Harnessing the representation divergence in unified multimodal models for mutual reinforcement. In *International Conference on Machine Learning (ICML)*, 2026. 3

Yinyi Luo, Wenwen Wang, Hao Bai, Marios Savvides, and Jindong Wang. LatentUMM: Dual latent alignment for unified multimodal models. *arXiv preprint arXiv:2605.17766*, 2026. 3

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594, 2023. 3

Wen Shi, Angela Yu, Runlin Fang, Hui Ren, Kezhi Wang, Andy Zhou, Chen Tian, Xuhai Fu, Yuqian Hu, Zi Lu, et al. Mathcanvas: Intrinsic visual chain-of-thought for multimodal mathematical reasoning. *arXiv preprint arXiv:2510.14958*, 2025. 1

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023. 3

Jiao Sun, Deqing Fu, Yushi Hu, Su Wang, Royi Rassin, Da-Cheng Juan, Dana Alon, Charles Herrmann, Sjoerd van Steenkiste, Ranjay Krishna, and Cyrus Rashtchian. DreamSync: Aligning text-to-image generation with image understanding feedback. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5920–5945, 2025. 3

Gladys Tyen, Hassan Mansoor, Victor Carbune, Peter Chen, and Tony Mak. LLMs cannot find reasoning errors, but can correct them given the error location. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 13894–13908, 2024. 2, 3

Haiyi Wang, Ao Su, Wei Ren, Feng Lin, and Wei Chen. Pixel reasoner: Incentivizing pixel-space reasoning with curiosity-driven reinforcement learning. *arXiv preprint arXiv:2505.15966*, 2025. 3

Shuai Wang, Liang Li, Yang Chen, Ruopeng Gao, Yao Teng, and Limin Wang. UniDDT: Unifying multimodal understanding and generation with decoupled diffusion transformer. *arXiv preprint arXiv:2606.16255*, 2026a. 3

Xinlong Wang, Zhengyang Zhang, Zheng Luo, Que Sun, Yufeng Cui, Jianyong Wang, Fan Zhang, Yue Wang, Zhen Li, Qixiang Yu, et al. Emu3: Next-token prediction is all you need. *arXiv preprint arXiv:2409.18869*, 2024. 1, 3

Yuqing Wang, Zhijie Lin, Ceyuan Yang, Yang Zhao, Fei Xiao, Hao He, Qibin Zhao, Zhengming Ding, and Fang Wang. Representation forcing for bottleneck-free unified multimodal models. *arXiv preprint arXiv:2605.31604*, 2026b. 3

Xinyu Wei et al. WEAVE: Unleashing and benchmarking the in-context interleaved comprehension and generation. *arXiv preprint arXiv:2511.11434*, 2025. 1, 3

Jialong Wu, Xiaoying Zhang, et al. Visual generation unlocks human-like reasoning through multimodal world models. *arXiv preprint arXiv:2601.19834*, 2026. 3, 7

Xueqing Wu, Yuheng Ding, Bingxuan Li, Pan Lu, Da Yin, Kai-Wei Chang, and Nanyun Peng. VISCO: Benchmarking fine-grained critique and correction towards self-improvement in visual reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9527–9537, 2025. 2, 3

Jinheng Xie, Wenhao Mao, Zechen Bai, Dujian Zhang, Wenxuan Wang, Kevin Qinghong Lin, Yuchao Gu, Zhenguo Chen, Zuwei Yang, and Mike Zheng Shou. Show-o: One single transformer to unify multimodal understanding and generation. In *International Conference on Learning Representations (ICLR)*, 2025. 1, 3

Qian Yang, Ankur Sikarwar, Huy Le, Le Zhang, and Zhuan Shi. How and what to imagine? visual thinking in unified multimodal models for cross-view spatial reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. 3

Ce Zhang, Zifu Wan, Zhehan Kan, Martin Q. Ma, Simon Stepputtis, Deva Ramanan, Russ Salakhutdinov, Louis-Philippe Morency, Katia Sycara, and Yaqi Xie. Self-correcting decoding with generative feedback for mitigating hallucinations in large vision-language models. In *International Conference on Learning Representations*, 2025. 3

Di Zhang, Junxian Li, Jingdi Lei, Xunzhi Wang, Yujie Liu, Zonglin Yang, Jiatong Li, Weida Wang, Suorong Yang, Jianbo Wu, Peng Ye, Wanli Ouyang, and Dongzhan Zhou. Critic-V: VLM critics help catch VLM errors in multimodal reasoning. *arXiv preprint arXiv:2411.18203*, 2024. 3

Guozhen Zhang, Xiangyu Qiu, Yutao Cui, Tao Song, Chao Li, Junzhe Li, Tao Huang, Xin Zhang, Yang Li, and Jian Wu. HYDRA-X: Native unified multimodal models with holistic visual tokenizers. *arXiv preprint arXiv:2606.13289*, 2026a. 1, 3

Hao Zhang, Jiaxin Qi, Zhijiang Tang, et al. Do you see what you draw? a semantic closed-loop framework for holistic evaluation of unified multimodal models. *arXiv preprint arXiv:2608.11907*, 2026b. 3

Zheng Zheng, Mingyang Yang, Jiacong Hong, Chunhui Zhao, Guangwei Xu, Lei Yang, Changjie Shen, and Xue Yu. Deepeyes: Incentivizing "thinking with images" via reinforcement learning. *arXiv preprint arXiv:2505.14362*, 2025. 3

A Implementation Details

A.1 Data

Data construction pipeline. Data construction starts from 182,384 GT trajectories in Zebra-CoT spanning 18 categories. In the first round, a fixed random seed and the `mixed_uniform_subset_v1` strategy produce 43,622 unique candidate trajectories. For a GT Interleaved CoT trajectory of length L, we form the eligible text and image target sets E_T and E_I and cap the number of candidate nodes at $K_{\max} = \min(|E_T| + |E_I|, \max(2, \lfloor L/3 \rfloor))$. We then sample a target count from $[2, K_{\max}]$ with a fixed seed and select targets that cover both modalities. Bagel-Zebra-CoT independently generates a one-step text or image continuation for each target node. At target step s, the model receives only the question and GT prefix $\mathrm{GT}[:s]$; generated candidates from other target nodes are never fed back into the context, and text and image targets use independent fixed seeds. Text candidates pass an initial review, blind verification, and packageability check: the first review tests whether the candidate can replace the current GT text node, blind verification identifies semantic, factual, visual, reasoning, or completion errors, and the packageability check verifies that the context required by the recovery branch is complete. Image candidates are compared with the GT state at the same step for equivalence in content, operation outcome, and reasoning state. Both annotation paths return structured error evidence, a correction request, and reflection text for constructing recovery trajectories.

Draw Error detection prompt

You are an expert in detecting errors in interleaved multimodal chains of thought. Given a partial trajectory, determine whether the newest image marked To Check contains a Draw Error: the current text proposes an executable action, but the image fails to realize the corresponding state transition. Earlier steps have already been validated, and the current text action is treated as correct and executable. Evaluate only the marked image against that action and the preceding validated visual state. Do not reassess the text policy. If the transition is faithfully realized, return `has_error: false`.

Assign exactly one of the three Draw Error types. **Instruction Mismatch** (`instruction_mismatch`) means that the image omits the requested operation, performs a different operation, applies it to the wrong target, or renders the wrong option, movement, annotation, or state change. **State Inconsistency** (`state_inconsistency`) means that the requested action is at least partly realized, but task-critical content inherited from the problem or preceding state is spuriously added, removed, moved, recolored, relabeled, or otherwise rewritten. **Rendering Error** (`rendering_error`) means that blank output, corruption, garbled content, artifacts, severe blur, or another rendering failure makes the image unusable for subsequent reasoning. Cosmetic differences that preserve the requested transition and task-critical state are not errors. If symptoms overlap, select the primary failure: use `rendering_error` when the content cannot be reliably assessed, otherwise `instruction_mismatch` when the requested transition itself is wrong, and `state_inconsistency` when the requested transition is present but carried-over state is corrupted.

Return plain JSON without Markdown fences. For an error, include `has_error`, `step`, `modality=image`, `error_type`, `error_description`, and `error_evidence` with `text_claim`, `visual_reality`, and `why_error`. Also return `correction_tool`, `images`, `correction_prompt`, and `reflect_text`. For a valid image, return `has_error: false`, the step and modality, and a short `reason`.

Use `image_edit` by default for reconstruction, state changes, restoring missing elements, removing hallucinations, or edits that combine content changes and annotations. Its correction prompt must be a concise, self-contained English instruction that names the base image, requested edit, and expected state. Use `visual_annotate` only for overlays such as lines, bounding boxes, arrows, points, or zoom crops. Its instruction must state the task, current state, error diagnosis, and exact annotation. When correcting a missing state transition with `image_edit`, use `problem_image_1` at the first step and preferably `reasoning_image_{k-1}` at later steps. Never use the faulty `reasoning_image_{k}` as the reconstruction base. Write `reflect_text` in the first person and in the language of the original reasoning trace.

Policy Error detection prompt

You are an expert in detecting errors in interleaved multimodal chains of thought. Given a partial trajectory, determine whether the newest text block marked To Check contains a Policy Error: an action proposed by the text that is not executable under the current validated visual state and the task rules. Earlier steps have already been validated. Judge local executability using only the problem, visible images, validated prefix, current text, and explicit task rules. Do not use a ground-truth answer, reference solution, correct next image,

hidden evaluation label, or unobserved future state. Do not assume that the marked text is erroneous. If it proposes an executable action, or no explicit next-state action, return `has_error: false`.

Assign exactly one of the three Policy Error types using the following mutually exclusive order. First, **Unavailable Target** (`unavailable_target`) means that the action refers to an object, attribute combination, or required quantity that is absent from the current state. Second, **Unmet Precondition** (`unmet_precondition`) means that the target exists, but a prior operation or state required by the action has not been established. Third, **Infeasible Action** (`infeasible_action`) means that the target and required prior state exist, but a spatial, task, physical, or action-rule constraint blocks the proposed transition. Report the earliest unexecutable action in the current block. A suboptimal but executable route, a different valid choice, disagreement with a reference trajectory, an incorrect final answer, or a general observation, reasoning, repetition, formatting, or completion defect is not by itself a Policy Error. Such a defect qualifies only when the visible current state and task rules prove that its proposed action is locally unexecutable; classify that action by the three definitions above.

Return plain JSON without Markdown fences. For an error, include `has_error`, `step`, `modality=text`, `error_type`, `error_description`, and `error_evidence` with `text_claim`, `visual_reality`, and `why_error`. Set `correction_tool` to `text_rewrite`, and return `correction_prompt`, `reflect_text`, and `corrected_text`. For a valid block, return `has_error: false`, the step and modality, and a short `reason`.

The correction prompt is self-contained and comprises four parts: task context, current reasoning state, error diagnosis, and requested revision. Include the problem images and `reasoning_image_{k-1}` whenever the current text depends on the preceding visual state. Preserve valid content and the original language. In the corrected text, replace only the unexecutable action and its directly dependent explanation, retain the `<think>` wrapper, and propose an action executable from the visible state without optimizing toward a hidden solution. The first-person reflection identifies the violated executability condition and states the corrected action in the language of the original trajectory. Every correction follows from the visible input and validated prefix rather than a reference answer.

Joint filtering and trajectory assembly. After final filtering, Draw Errors and Policy Errors are joined by source Parquet file and source row. The first round contains 18,217 trajectories with valid Draw Errors and 13,822 with valid Policy Errors. Their union produces 21,053 recovery trajectories, including 10,986 with both error types. Each source sample is reconstructed once, with all accepted errors written in their original step order. The two recovery branches have the following structures.

Policy Error branch

`Existing GT context → ERROR THOUGHT: model-generated erroneous text → visual reflection signal → THOUGHT: GT corrected text at the current step → subsequent GT image`

Draw Error branch

`Existing GT context → error_image_k: model-generated erroneous image → THOUGHT: reflection and redraw instruction → GT image at the current step`

Dataset composition. Zebra-CoT contains 182,384 source trajectories across 18 categories. The first round samples 43,622 candidate trajectories and constructs 21,053 training trajectories containing text reflection, image reflection, or both. These trajectories contain 38,825 error nodes: 22,219 Draw Errors and 16,606 Policy Errors, or 1.844 error nodes per trajectory on average. In total, 84.78% of the trajectories contain one or two error nodes, and the maximum is eight.

Table 5: Error-node distribution in the first-round recovery dataset.

Modality	Statistic or error type	Nodes	Within-modality share
Image	**All Draw Errors**	**22,219**	**100%**
Image	Instruction Mismatch	12,205	54.93%
Image	State Inconsistency	9,594	43.18%
Image	Rendering Error	420	1.89%
Text	**All Policy Errors**	**16,606**	**100%**
Text	Unavailable Target	8,263	49.76%
Text	Infeasible Action	829	4.99%
Text	Unmet Precondition	7,514	45.25%
All	**All error nodes**	**38,825**	–

Table 6: Category distribution of the first-round recovery dataset. "Sel." denotes the fraction of source trajectories selected, and "Share" denotes the category's fraction of the final dataset.

Category	Source	Final	Sel. (%)	Share (%)	Joint	Draw-only	Policy-only	Draw Errors	Policy Errors
2D Visual Reasoning – Visual Jigsaw	21,899	1,683	7.69	7.99	616	874	193	1,715	883
2D Visual Reasoning – Visual Search	30,000	2,178	7.26	10.35	847	901	430	1,748	1,277
3D Visual Reasoning – Embodied CoT	22,666	3,567	15.74	16.94	2,368	764	435	4,259	3,865
3D Visual Reasoning – Multi-Hop Objects Counting	10,000	1,336	13.36	6.35	730	422	184	1,445	1,155
3D Visual Reasoning – Robot Planning	6,944	976	14.06	4.64	543	299	134	1,074	827
Scientific Reasoning – Chemistry	4,666	374	8.02	1.78	53	276	45	428	102
Scientific Reasoning – Competitive Programming	1,207	73	6.05	0.35	2	47	24	50	26
Scientific Reasoning – Geometry	1,058	222	20.98	1.05	169	40	13	209	182
Scientific Reasoning – Graph Algorithms	10,000	1,178	11.78	5.60	681	425	72	1,558	927
Scientific Reasoning – Physics	7,090	911	12.85	4.33	463	262	186	725	649
Visual Logic & Strategic Games – ARC-AGI	2,000	448	22.40	2.13	332	78	38	489	389
Visual Logic & Strategic Games – Checkers	2,753	604	21.94	2.87	474	70	60	712	861
Visual Logic & Strategic Games – Chess	20,483	2,955	14.43	14.04	1,685	845	425	2,940	2,484
Visual Logic & Strategic Games – Ciphers	6,589	628	9.53	2.98	167	418	43	955	231
Visual Logic & Strategic Games – Connect Four	2,029	461	22.72	2.19	354	59	48	498	544
Visual Logic & Strategic Games – Maze	20,000	2,051	10.26	9.74	963	711	377	1,870	1,499
Visual Logic & Strategic Games – RPM	3,000	413	13.77	1.96	211	141	61	352	272
Visual Logic & Strategic Games – Tetris	10,000	995	9.95	4.73	328	599	68	1,192	433
Total	**182,384**	**21,053**	**11.54**	**100.00**	**10,986**	**7,231**	**2,836**	**22,219**	**16,606**

A.2 TRAINING

Node-level loss masking and reflection dropout. All trajectories use Eq. (4), with node-level masks selecting the supervised positions. Draw Error images and Policy Error text remain in context but receive zero generation loss. Reflect-Draw text and corrected text nodes receive cross-entropy supervision, while correct images, visual reflection signals, and recovered images receive flow-matching supervision.

Table 7: Node-level loss masks for Self-Corrective Interleaved Chain of Thought. Reflect-Draw text is the recovery block following a Draw Error. The visual reflection signal is the fixed image following a Policy Error. FM is implemented as mean-squared error on the flow-velocity target.

Node type	In context	CE	FM
Correct text	✓	1	–
Correct image	✓	–	1
Draw error image I_i^-	✓	–	**0**
Reflect-Draw text	✓	**1**	–
Policy error text T_i^-	✓	**0**	–
Visual reflection signal I^{ref}	✓	–	**1**
Recovered $T_i^{\mathrm{gt}}/I_i^{\mathrm{gt}}$	✓	**1**	**1**

During data loading, we apply reflection dropout by replacing reflection nodes with their corresponding normal GT nodes. The dropout probabilities are 0.10 for Text-Reflect nodes and 0.15 for Draw-Reflect nodes.

Iterative error collection and replay. Let $M_0 = M_{\text{base}}$ denote the initial interleaved text–image model and M_k the checkpoint after training round k. Training uses $K = 3$ rounds of 1,500 steps each, for 4,500 steps in total. Let $\mathcal{B}_1$ denote the first-round recovery trajectories and $\mathcal{E}_k$ the recovery trajectories mined in round $k \in \{2, 3\}$.

In each round, the model generates text and image candidates from GT prefixes, and the VLM judge retains erroneous candidates and constructs their recovery trajectories. To preserve the model's Interleaved CoT capability, we sample 20% of Zebra-CoT in proportion to its category distribution, yielding $\mathcal{D}_{\text{keep}}$, and include it throughout training. Starting from the second round, the historical replay set $\mathcal{R}_k$ randomly samples 30% of the `joint` trajectories from the preceding recovery set $\mathcal{B}_{k-1}$. Before being mixed with $\mathcal{D}_{\text{keep}}$, both sets undergo reflection dropout: Text-Reflect and Draw-Reflect nodes are independently replaced by their corresponding normal GT nodes with probabilities 0.10 and 0.15, respectively.

$\mathrm{StratifiedSample}$ denotes category-proportional sampling, $\mathrm{ReflectDropout}$ applies this node replacement at the specified probabilities, and Mix forms the training mixture. $\mathrm{SFT}(M, \mathcal{D}; \mathcal{L})$ denotes 1,500 steps of supervised fine-tuning with Eq. (4). Algorithm 1 summarizes the schedule.

Algorithm 1 Iterative error replay for Self-Corrective Interleaved Chain of Thought

Require: Base model M_{base}; Zebra-CoT $\mathcal{D}_{\text{Zebra}}$
Require: First-round recovery set $\mathcal{B}_1$; rounds $K = 3$
$\mathcal{D}_{\text{keep}} \leftarrow \mathrm{StratifiedSample}(\mathcal{D}_{\text{Zebra}}, 0.20)$
$\widetilde{\mathcal{B}}_1 \leftarrow \mathrm{ReflectDropout}(\mathcal{B}_1; p_T = 0.10, p_D = 0.15)$
$M_1 \leftarrow \mathrm{SFT}\Big(M_{\text{base}}, \mathrm{Mix}(\mathcal{D}_{\text{keep}}, \widetilde{\mathcal{B}}_1); \mathcal{L}\Big)$ for 1,500 steps
for $k = 2, \ldots, K$ **do**
  $\mathcal{E}_k \leftarrow \varnothing$
  Sample GT trajectories and choose a random text or image point at step i
  Generate from each GT prefix with M_{k-1}; obtain a VLM judgment
  if the generated node is correct **then**
    Discard the generated node
  else if the incorrect node is an image **then**
    Add τ_i^{draw} from Eq. (2) to $\mathcal{E}_k$
  else
    Add τ_i^{policy} from Eq. (3) to $\mathcal{E}_k$
  end if
  $\mathcal{R}_k \leftarrow \mathrm{Sample}(\mathcal{B}_{k-1}^{\text{joint}}, 0.30)$
  $\widetilde{\mathcal{C}}_k \leftarrow \mathrm{ReflectDropout}(\mathcal{E}_k \cup \mathcal{R}_k; p_T = 0.10, p_D = 0.15)$
  $\mathcal{D}_k \leftarrow \mathrm{Mix}\Big(\mathcal{D}_{\text{keep}}, \widetilde{\mathcal{C}}_k\Big)$
  $M_k \leftarrow \mathrm{SFT}(M_{k-1}, \mathcal{D}_k; \mathcal{L})$ for 1,500 steps
  $\mathcal{B}_k \leftarrow \mathcal{E}_k$
end for
return M_K

Training configuration. We initialize from Bagel-Zebra-CoT and jointly fine-tune on current-round recovery data, capability-preserving data, and historical replay. Training uses eight NVIDIA RTX PRO 6000 GPUs with BF16 precision and FSDP `FULL_SHARD`.

A.3 Inference

Inference uses BF16 precision with sampling enabled (`do_sample=true`), a text temperature of 0.3, text and image CFG scales of 4.0 and 2.0, a CFG interval of $[0, 1]$, a timestep shift of 3.0, 50 diffusion steps, and a random seed of 42. Starting from the question and problem image, the model alternates between text and image thoughts and appends every generated node to the context until the output contains `Final Answer:` or `<answer>`. After an erroneous visual thought, the model can emit reflection text and redraw the image at the same logical step. After an invalid action, it can emit a visual reflection signal and retry the action from the latest valid state. This process runs

Table 8: Training configuration.

Parameter	Setting
Training data	Current-round Zebra-CoT-Reflect recovery data; 20% category-proportional sample from Zebra-CoT for capability preservation; 30% of the preceding round's `joint` recovery trajectories for historical replay
Reflection dropout	Text-Reflect 0.10; Draw-Reflect 0.15
Hardware	8× NVIDIA RTX PRO 6000
Precision and parallelism	BF16; FSDP `FULL_SHARD`
Learning rate	2×10^{-5}
Scheduler	Cosine decay; 50 warm-up steps; minimum learning rate 10^{-6}
Training steps	1,500 steps per round; 3 rounds; 4,500 steps in total
Loss weights	CE 1.0; MSE 1.0
Maximum batch tokens	60,000
Conditional dropout	Text 0.1; ViT 0.3; VAE 0.3
Frozen component	VAE

within the standard Interleaved CoT sequence without an external error detector, recovery counter, or separate routing interface.

System prompt

You are an AI reasoning assistant capable of step-by-step Interleaved Chain of Thought with language and visual thoughts. Think step by step and use visual aids to enhance your problem-solving. Provide your final conclusion clearly in the format of "Final Answer: ¡answer here¿"

B Diagnosing Maze and Sokoban Failures

Diagnostic setup. To isolate the low grid-planning scores, we inspect representative failures and test four controlled inference settings. Open-loop Interleaved CoT generation lets BAGEL-Zebra-CoT produce both text and visual states without external execution. Text-corrected generation uses Qwen3.7-Plus to revise coordinates and action descriptions before BAGEL-Zebra-CoT generates the next image. Direct route generation asks Qwen3.7-Plus to predict all Maze waypoints from the input image in one pass. Environment-grounded execution applies one atomic action at a time, renders the resulting state, and constrains the next action to the current shortest path or optimal solution. Table 9 summarizes the resulting failure patterns.

Table 9: Diagnostic tests for grid reasoning. The first row reports the main VisWorld-Eval results. The remaining settings probe state persistence and transition legality. Environment-grounded execution receives rendered states and shortest-path or optimal-action constraints, so its solved rates measure performance under this controlled assistance.

Setting	State source	Transition control	Result or observed failure
SC-CoIT main evaluation	Generated Interleaved CoT state	Learned local reflection	Maze 0.00%, Sokoban 0.83%
Open-loop generation	BAGEL-Zebra-CoT text and images	None	Coordinates and object states drift
Text-corrected generation	Qwen3.7-Plus text, BAGEL-Zebra-CoT images	None	Correct endpoints still yield illegal edges
Direct route generation	Qwen3.7-Plus from the input image	No intermediate feedback	Start 98.59%, goal 91.55%, legal route 8.45%
Environment-grounded execution	Qwen3.7-Plus and environment-rendered state	Optimal legal action at each step	Maze completion 99.08%, Sokoban completion 85.05%

Both tasks require a persistent structured state. Maze reasoning must preserve the wall-constrained connectivity of the grid, the current player location, the goal, and the remaining route. Each move

must stay within the grid, reach an adjacent cell, and avoid crossing a wall. Sokoban additionally requires the model to preserve box and target locations, distinguish movement from pushing, update the player and box together, and reject blocked pushes, box pulling, and deadlocking moves.

Maze: localization without topology. The direct-route test separates endpoint detection from connectivity. Among predictions that could be mapped reliably to grid cells, start and goal localization reach 98.59% and 91.55%, respectively, but only 8.45% pass both legality and goal-reachability checks. The text-corrected example in Figure 5(a–c) likewise identifies the start at `[2,0]` and goal at `[0,1]`, yet proposes the blocked move from `[2,0]` to `[2,1]` and draws through a wall near the goal. Correct coordinates therefore do not supply the wall-constrained connectivity needed for search.

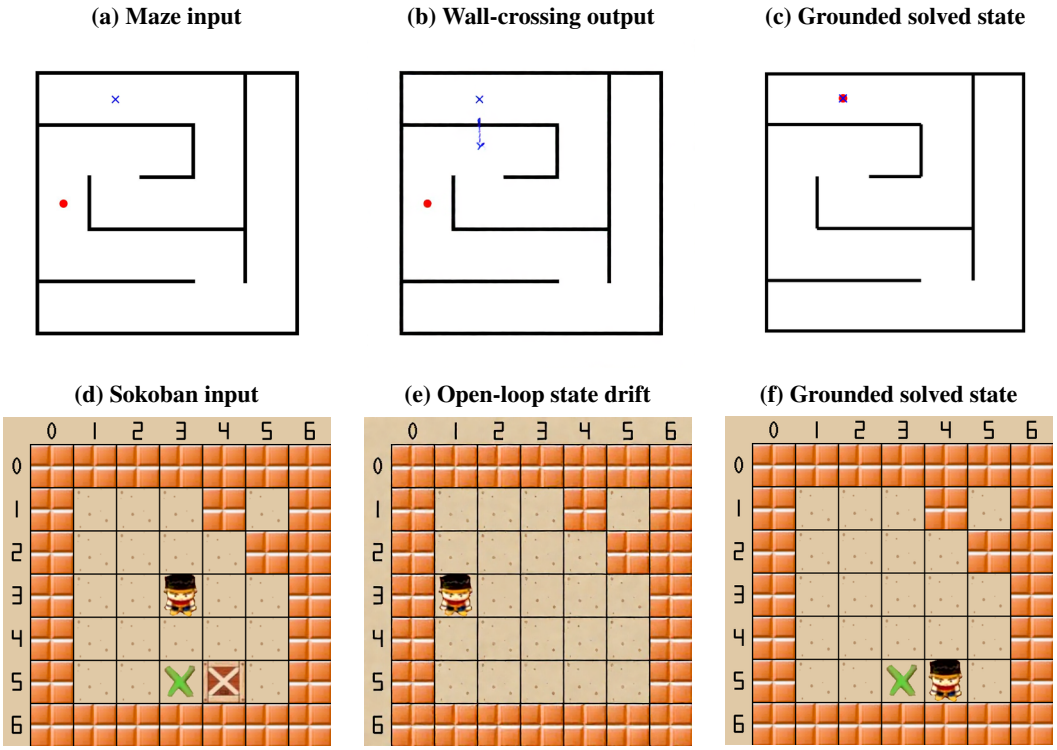

Figure 5: Representative state failures and controlled recovery. In Maze, corrected endpoint coordinates still produce a mark that crosses the upper wall, whereas environment-rendered execution reaches the goal through legal transitions. In Sokoban, an open-loop visual state drops the box and target while moving the player inconsistently. Grounded execution maintains the discrete state until the box reaches the target. The solved Sokoban renderer displays the target marker; in the logged terminal state, the box and target both occupy `[5,3]`.

Sokoban: dynamic state without executable transitions. Sokoban adds moving objects and push preconditions to the same topology problem. In Figure 5(d–f), open-loop reasoning first proposes moving to the side of the box, but the generated states displace the player and remove both the box and target. Once these objects disappear from the visual state, a later "push" has no executable meaning. With environment-rendered states, the verified sequence `right, right, down, down, left` ends with the player at `[5,4]` and the box on the target at `[5,3]`.

Takeaway. Under environment-grounded execution, Maze and Sokoban reach 99.08% and 85.05% solved-state accuracy, respectively. These results locate the primary failure before and during search: the model must reconstruct wall topology, preserve dynamic object state, and validate every atomic transition. SC-CoIT repairs local text–image errors at modality boundaries, but its current inference loop does not explicitly reconstruct wall adjacency or enforce transition legality. Small state mismatches can thus accumulate until a long trajectory no longer corresponds to

the board. Persistent discrete state and transition validation are therefore complementary to learned cross-modal recovery on grid-planning tasks.

C CASE STUDIES

Each case below presents the problem and the recorded local recovery sequence on one slide. Teal emphasis identifies requested changes or recovered content, and coral emphasis identifies an error or an unchanged incorrect result. Quoted text is excerpted from the trajectory.

Draw reflection: successful repair. In the Multi-Hop case in Figure 6, the question asks the model to apply a sequence of object edits and then resolve a spatial relation. Before performing those edits, the model requests a Right45 view of the initial scene. This operation should change the viewpoint while preserving the objects, but the generated image drops the red cylinder and contains only three objects. The reflection names the missing cylinder, returns to the original problem image as the reference, and explicitly lists the four objects that the redraw must retain. The next image restores the cylinder alongside the other three objects. The trajectory subsequently reaches answer B, matching the reference.

The repair matters because the intermediate image carries the entities on which later operations depend. Omitting an object changes the task state even when the remaining scene looks plausible. Here, the reflection turns a visual discrepancy into a concrete correction by specifying both what to restore and what to preserve. The case illustrates how Draw reflection can recover task-relevant evidence before subsequent reasoning uses an incomplete scene.

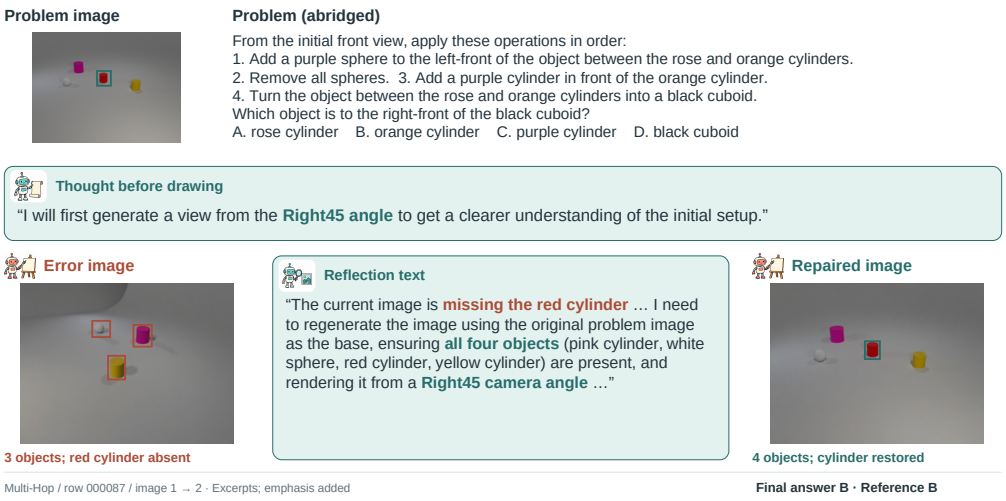

Figure 6: **Draw reflection: successful repair in Multi-Hop.** The problem, intended drawing, error image, reflection text, and repaired image form one recorded recovery sequence.

Draw reflection: unsuccessful repair. In Figure 7, the Cube 3-View question asks how many green cubes can possibly be seen from the left. The initial thought requests that view, but the generated image reproduces the supplied front-right, right, and top views. The reflection identifies the mismatch and repeats the required Left perspective. Yet the next image retains the same three views, as shown by both the depicted projections and their labels. The trajectory ends with A (1) instead of the reference B (5). The requested observation is thus absent both before and after reflection, despite an explicit and relevant correction.

Compared with the preceding object-restoration case, this repair requires a geometric transformation that changes which cube faces are visible while preserving the stack. The text specifies the correct target, but the image generator does not realize it. This local separation explains why diagnosis

quality and repair success must be evaluated independently, consistent with Section 4.4. It also identifies a concrete target for stronger repair supervision: the next image must execute the requested viewpoint change, with object layout and visibility checked against that target. Successful repair therefore requires the relevant visual state to change in accordance with the correction.

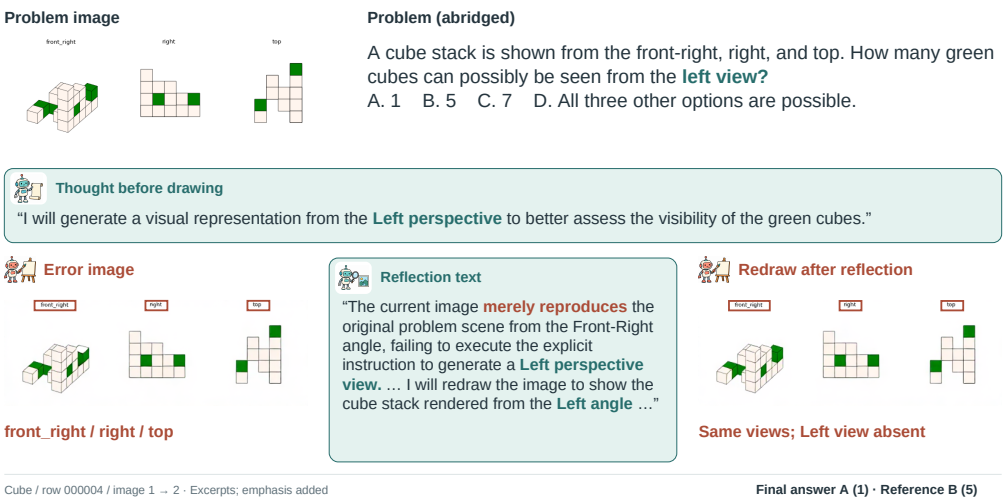

Figure 7: **Draw reflection: unsuccessful repair in Cube 3-View.** Highlighted labels show that the requested Left view remains absent after an explicit redraw instruction.

Policy reflection: successful repair. The Paper Folding case in Figure 8 asks for the number of upward triangles minus the number of circles. With the preceding generated visual state in context, the text before reflection announces that it will count the shapes but supplies neither count nor the requested difference. After reflection, the continuation reports six upward triangles and one circle, explicitly computes $6 - 1 = 5$, and gives the reference answer. The change is substantive: an intention to perform the operation becomes an executed calculation with intermediate quantities and a result.

This case makes the recovery target easy to identify from the question and the unfinished text. The pending operation is already known, so the next text node can repair the omission by carrying it through. It illustrates a useful role for Policy reflection in enforcing task completion: progress should be judged by whether the requested computation has been performed, rather than by whether the continuation fluently describes what it plans to do. The explicit counts also make the repaired step inspectable, connecting the model's reported visual interpretation to the final arithmetic.

Policy reflection: unsuccessful repair. In the Cube 3-View case in Figure 9, the question asks how many green cubes can possibly be seen from the back, while the supplied views are front-left, left, and front. The displayed continuation starts from a generated visual state and concludes A (1). After reflection, the next text again gives A (1), with no revised count or new visibility argument; the reference is C (4). The answer has been restated as final, but the underlying decision remains unchanged.

The contrast with Paper Folding distinguishes completing an omitted operation from revising an incorrect judgment. In the successful case, the continuation fills an explicit computational gap. Here, there is already a definite answer, and recovery requires reconsidering which cubes the requested back view would expose. Further continuation leaves the error intact when it reuses the same visibility judgment. This suggests that useful Policy repair should make the disputed decision explicit and re-evaluate it against the task evidence. For this case, the meaningful repair target is the back-view count and its spatial justification, not another statement of the existing answer.

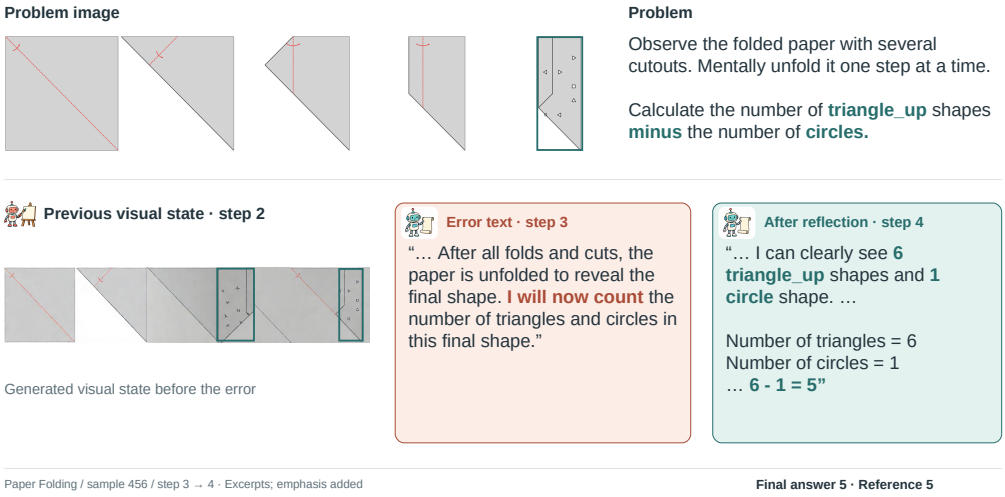

Figure 8: **Policy reflection: successful repair in Paper Folding.** The previous generated visual state accompanies the incomplete text and the continuation that supplies the calculation.

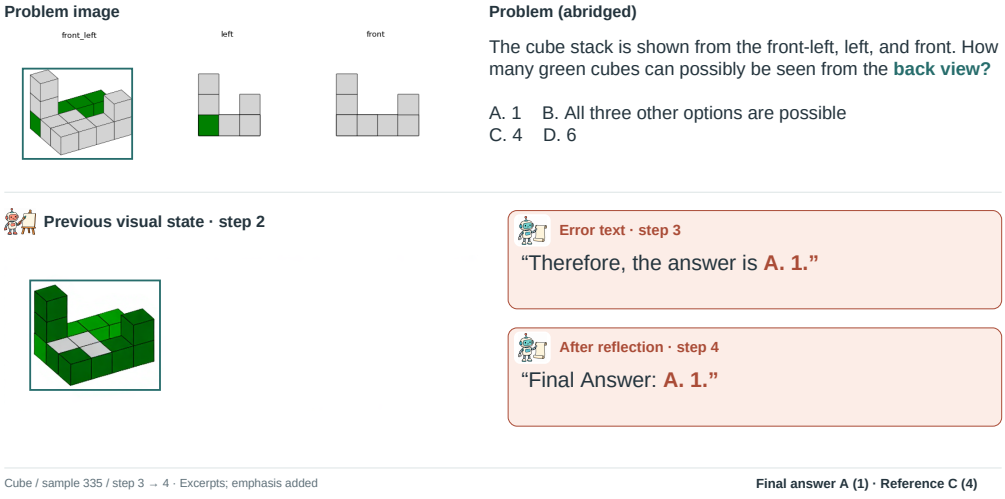

Figure 9: **Policy reflection: unsuccessful repair in Cube 3-View.** The generated visual state precedes two text nodes that retain the same incorrect count before and after reflection.

Cross-case insight. Across the four cases, success depends on whether the next node makes the required task-relevant change: restoring an omitted object, executing a requested view change, completing a pending calculation, or revising a visibility judgment. Draw reflection repairs the evidence available to reasoning, while Policy reflection repairs the operation or decision made from that evidence. Their shared repair bottleneck is the translation of feedback into that specific change. These cases motivate supervision that links an identified error to its corrected state, and evaluation that checks whether the requested change actually occurs while preserving valid context.